



Japanese Laid-Open Patent Application No. 2001-92661

[0086] FIG. 10 shows parallel process selection method used in a source program of the CPU1 and the CPU1A.

[0087] A C-language source program undergoes processing by a preprocessor before being compiled. In a source program, the preprocessor performs processing using a preprocessor statement indicated as "#", and an instruction to perform compiling is then given to a compiler.

[0088] FIG. 10 shows a source program written in C language in which it is possible to instruct the compiler to perform compiling using the preprocessor statement (#pragma). In the case of (a), the function "kansu1" and the function "kansu2" are compiled two in parallel and four in parallel respectively, where

#pragma parallel4 (kansu1) and

#pragma parallel2 (kansu2)

[0089] Furthermore, in the case of (b), the descriptions in between #pragma parallel2 and #pragma parallel2end are compiled two in parallel (in this case, a while loop).

[0090] As similar to the case of the CPU1 shown in FIG. 1, in the case where four processing details are described in an instruction code of 128 bits, when the instruction code is compiled two in parallel, fields corresponding to two predetermined arithmetic processing blocks are not manipulated. Using an instruction decode block DU, the no-operation fields cause a stop signal STPi to be asserted on the corresponding arithmetic processing blocks so as to stop the arithmetic processing manipulation.

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2001-092661

(43)Date of publication of application : 06.04.2001

(51)Int.Cl.

G06F 9/38
 G06F 1/32
 G06F 1/04
 G06F 9/30
 G06F 9/45

(21)Application number : 11-267950

(71)Applicant : HITACHI LTD

(22)Date of filing : 22.09.1999

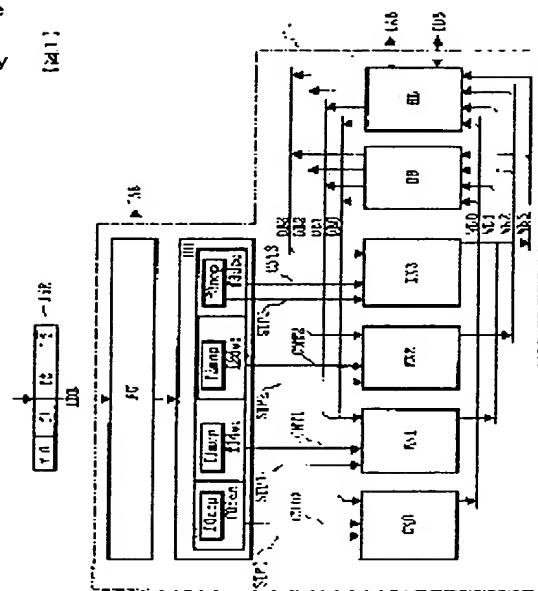
(72)Inventor : MITSUISHI NAOMIKI

(54) DATA PROCESSOR

(57)Abstract:

PROBLEM TO BE SOLVED: To provide a data processor for controlling the parallelism of parallel processing, and for easily operating power consumption control corresponding to the parallelism without increasing any logical scale.

SOLUTION: A CPU 1 capable of parallel arithmetic processing by using plural executing parts (EX0-EX3) by decoding read instructions stops the operation clock signal of any executing part in a non-operational state, and inhibits any data input or output when the number of arithmetic processing to be executed in parallel is smaller than the number of executing parts at the time of operating the parallel arithmetic processing. In this case, an instruction code itself to designate the contents of the arithmetic processing at the executing parts is provided with information indicating whether or not the processing should be executed in parallel. Thus, it is possible to reduce any undesired power consumption as necessary.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁 (J P)

(12) 公 開 特 許 公 報 (A)

(11)特許出願公開番号

特開2001-92661

(P2001-92661A)

(43)公開日 平成13年4月6日(2001.4.6)

(51)Int.Cl. ⁷	識別記号	F I	テ-マ-コ-ト*(参考)	
G 0 6 F	9/38	3 7 0	3 7 0 A	5 B 0 1 1
	1/32		3 0 1 C	5 B 0 1 3
	1/04	3 0 1	3 3 0 B	5 B 0 3 3
	9/30	3 3 0	3 5 0 F	5 B 0 7 9
		3 5 0	3 3 2 B	5 B 0 8 1
		1/00		

審査請求 未請求 請求項の数8 O L (全 16 頁) 最終頁に続く

(21)出願番号 特願平11-267950

(22)出願日 平成11年9月22日(1999.9.22)

(71)出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目6番地

(72)発明者 三ッ石 直幹

東京都小平市上水本町五丁目20番1号 株式会社日立製作所半導体グループ内

(74)代理人 100089071

弁理士 玉村 静世

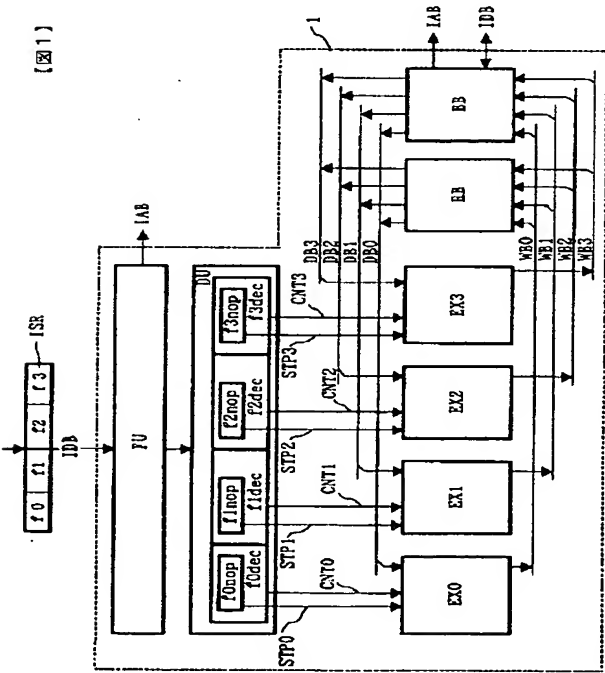
最終頁に続く

(54)【発明の名称】 データ処理装置

(57)【要約】

【課題】 並列処理の並列度が制御可能であってその並列度に応じた消費電力制御を、論理的規模を増大させず、容易に行なえるデータ処理装置を提供する。

【解決手段】 読み込んだ命令を解説し複数の実行部 (E X 0 ~ E X 3) を用いて並列演算処理可能な C P U (1) は、並列演算処理を行なう場合に、実行部の数より、並列実行すべき演算処理が少ないとき、動作しない実行部の動作クロック信号を停止すると共にデータ入出力などを禁止する。実行部での演算処理内容を指定する命令コードそれ自体に、並列実行すべき処理であるか否かを示す情報を持たせる。これにより、不所望な電力消費を、随時、抑制することができる。



【特許請求の範囲】

【請求項1】 並列処理を行なうデータ処理装置であって、

命令をリードする命令フェッチ部と、並列動作可能であって夫々演算器を内蔵した複数の実行部と、前記命令フェッチ部がリードした命令を解釈して得られる制御信号に基づいて前記複数の実行部の動作を制御する制御部と、を有し、

前記夫々の実行部は、前記制御部の出力する所定の制御信号にตอบสนองして個別的に動作停止状態を採り得るものであることを特徴とするデータ処理装置。

【請求項2】 前記制御部は、命令コード中に含まれ、前記実行部の処理を指定する情報のフィールドに対し、前記処理の無操作を判定したとき、対応する実行部の動作を停止状態に制御する前記所定の制御信号を生成するものであることを特徴とする請求項1記載のデータ処理装置。

【請求項3】 前記制御部は、所定の命令コードを解釈したとき、これに続く命令の実行を並列させる所定の制御信号を複数の実行部に出力するものであることを特徴とする請求項1記載のデータ処理装置。

【請求項4】 演算動作停止させる一部の実行部を指定する手段を更に有し、前記制御部は、その手段で指定された実行部を、命令の並列実行対象から除外するものであることを特徴とする請求項3記載のデータ処理装置。

【請求項5】 前記指示する手段は前記実行部によってアクセスされるレジスタ手段であることを特徴とする請求項4記載のデータ処理装置。

【請求項6】 前記実行部は、第1のクロック信号に同期動作される演算器を有し、前記制御部からの所定の制御信号にตอบสนองして前記第1のクロック信号の変化を抑止して演算動作を停止させるゲート手段を有して成るものであることを特徴とする請求項1乃至5の何れか1項に記載のデータ処理装置。

【請求項7】 異なる分周比のクロック信号を生成する分周器と、分周器の出力を選択するセレクタと、セレクタによる選択の制御情報が設定されるクロック選択レジスタとを有し、

前記第1のクロック信号は前記セレクタで選択されたクロック信号に基いて生成されるものであることを特徴とする請求項6記載のデータ処理装置。

【請求項8】 請求項1乃至7の何れか1項に記載のデータ処理装置が実行するオブジェクトプログラムを生成する開発装置であって、ソースプログラムのステップ関数またはファイルの少なくとも一つの単位で前記実行部の並列度を指定する手段を有し、これによって指定された並列度を前記単位でオブジェクトプログラムに反映するものであることを特徴とする開発装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、マイクロコンピュータ、マイクロコントローラ、データプロセッサ等のデータ処理装置に関し、例えば、単一の半導体チップに形成された、機器制御用（機器組込み用途）のデータ処理装置に利用して有効な技術に関するものである。

【0002】

【従来の技術】機器制御用のマイクロコンピュータには、所謂リアルタイム性が要求される。即ち、制御対象機器の制約から、所定のイベントに呼応した処理を、所定の時間内に完了し、制御を行なう必要がある。また、システム全体の時間的な制約の中で、所要の処理を完了させる必要がある。更に、イベントは同時に発生する場合もあるから、このような場合、各イベントに呼応した処理に対して、夫々の制約を満足しなければならない。データ処理全体の処理時間ではなく、イベント毎に、個別の処理の時間を短縮する必要があるから、例えば、1個のCPUによるような、単一の命令ストリームに対応する処理性能を向上させる必要がある。

【0003】例えば、OS（オペレーティングシステム）を使用する場合には、OSの処理プログラム実行中には、割込みが禁止されることが一般的であるが、これは、イベントに対する処理開始を遅らせることになるから、機器制御やリアルタイム処理においては好ましくない。このため、機器制御用のOSにはリアルタイムOSなどと呼ばれる、割込み禁止時間を短縮したり、所要の優先順位の割込みは常に受け付けられるようにしたものを用いられる。

【0004】データ処理装置において、単一の命令ストリームに対応するように、1個のCPUで、いわゆる並列処理を行ない、実行時間を短縮可能にするものとして、平成7年1月（株）日経BP社発行「日経エレクトロニクス」pp68～80「1998年に転機、ハードウェアを単純化してVLIWへ」などに記載されているように、スーパスカラやVLIW（Very Long Instruction Word）などがある。いずれも演算処理ブロックを複数持ち（例えば4個）、同時に実行できる処理の数を増やして（例えば4並列）、全体的な処理性能を向上している。

【0005】前記VLIWは、128ビットなどの長い固定長の命令コードを持ち、演算処理ブロックに夫々対応する各フィールドに、処理内容を指定するようになっている。コンパイル時に、ソースプログラムを解析して、予め並列実行できる処理を、1つのVLIWの命令に構成する。

【0006】一方、スーパスカラは、実行時に、リードした個々の命令を解析して、並列実行可能な命令を選択して、並列処理を行なう。この場合、コンパイル時に、ソースプログラムを解析して、並列処理を制御する命令を挿入し、この制御命令に基づいて、続く複数命令を並列処理する例もある。

【0007】

【発明が解決しようとする課題】しかしながら、上記の並列処理では、複数の処理を、常に実行可能であるとは限らない。例えば、データの依存関係によるものがあり、所定のライトの後にリードを行なうべき場合に、並列処理を行なうと、この順序を保てなくなることなどがあるからである。並列処理を行なえない場合には、一部の演算処理ブロックは使用されない。

【0008】また、データ処理装置の応用分野は広がっているから、携帯型などバッテリーで動作するようなものなど、処理速度とともに、低消費電力が必要になるものもあり、実使用上、常に最大の処理速度が必要であるとは限らない。

【0009】消費電力を低減する為に、CPU内部の機能ブロックのうち、当該命令で使用する或いは使用しない機能ブロックへのクロック信号供給を許可／禁止する例が、特開平10-340127号公報に記載されている。この例では、使用状態に応じて、処理速度を優先したり、消費電力を優先したりすることには対応できない。また、命令や処理速度などについては言及されていない。

【0010】一方、特開平9-185589号公報には、複数のタスクを並列に処理可能な複数のプロセッサを有し、タスクを構成する処理単位毎のプロセッサ資源の要求に応答してタスク管理手段が、休止状態のプロセッサを発生させるように、複数のプロセッサの内の特定のプロセッサにプロセッサ資源の不足が生じない範囲でタスクを割り付け、電源供給制御手段により、休止状態のプロセッサに対する電源供給を停止し、システム全体の消費電力を節減可能にしたデータ処理システムが例示されている。

【0011】この例では、各タスクが個別のプロセッサに割当てられるから、各タスクの処理時間は、プロセッサ全体の休止状態には依存せず、一定である。複数のプロセッサは、所要の処理を高速化する為に用いられてはいない。

【0012】また、回路の一部に電源の供給を停止するような場合は、電源が供給されている部分と、供給されていない部分のインタフェース信号に、電氣的に不都合が生じないように、例えば、貫通電流が生じたり、トランジスタに不所望な電圧印加状態が生じたりしないようにする必要があり、そのために必要な回路構成によって論理的規模の増加を招く。特に、リアルタイム性が必要とされる場合には、必要に応じて、即座に動作可能でなければならない。

【0013】本発明の目的は、並列処理の並列度が制御可能であって、その並列度に応じた消費電力制御を、論理的規模を増大させず、容易に行なえるデータ処理装置を提供することにある。

【0014】本発明の前記並びにその他の目的と新規な

特徴は本明細書の記述及び添付図面から明らかになるであろう。

【0015】

【課題を解決するための手段】本願において開示される発明のうち代表的なものの概要を簡単に説明すれば下記の通りである。

【0016】〔1〕読み込んだ命令を解読して複数の実行部を用いて並列演算処理可能なCPU等のデータ処理装置において、並列演算処理を行なう場合に、複数の実行部は命令を解読して制御信号を生成する制御部の指示に応答して動作し、制御部の指示によって個別的に演算動作の停止が指示されるようになっている。動作停止が指示された実行部は、例えば、同期動作のクロック信号の変化が抑止されて、その演算動作が停止され、且つデータ入出力が禁止される。

【0017】上記の如く、並列処理を行なう場合に、動作しない実行部のクロック信号を停止すると共に、データ入出力などを禁止することにより、不所望な電力消費を抑制することができる。

【0018】その場合、実行部での演算処理内容を指定する命令コードそれ自体に、並列実行すべき処理であるか否かを示す情報を持たせる。また、ハードウェアで、並列処理可能な命令を検出している場合には、この検出結果を利用して、並列実行すべき処理であるか否かを検出する。

【0019】このように、並列実行すべき処理を検出するために、命令コード自体に、その情報を持たせ、また、ハードウェアで、並列処理可能な命令を検出している場合には、この検出結果を利用することにより、随時、消費電力を抑制できる。

【0020】〔2〕更に、消費電力を制御するために、動作許可する並列処理の数を指定可能にする。これにより、一部の実行部の動作を停止させて、消費電力を低減することができる。

【0021】具体的には、コンパイラで、使用する並列処理の数を指定可能にし、これに従って、コンパイラは命令コードを生成するようにするとよい。即ち、指定された数より大きい並列処理を実行する命令コードを生成しないようにする。これにより、容易に並列度を指定可能にし、使い勝手を向上することができる。

【0022】また、ハードウェアで、並列処理可能な命令を検出している場合には、制御レジスタによって使用可能な並列処理の数を指定して、前記ハードウェアはこれを参照して、並列処理の制御を行なうとよい。これにより、プログラムを変更することなく、随時、消費電力の制御を可能にして、使い勝手を向上することができる。プログラムを不所望に変更しないことによって、プログラムの開発効率を向上することができる。

【0023】〔3〕更には、動作クロック信号を可変にし、少なくとも、並列処理を行なわない程度まで、動作

クロック信号を低下させるようにするとよい。

【0024】動作クロック信号を可変にする場合には、周辺機能のクロック信号を固定にしつつ、データ処理装置などのバスマスタのクロック信号を変更するようにする。これにより、クロック信号切替え時の処理を最小限にすることができる。

【0025】

【発明の実施の形態】図1にはデータ処理装置の第1の構成例が示される。同図に示されるデータ処理装置はたとえばCPU（中央処理装置）1であり、命令リードブロックFU、命令デコードブロックDB、4個の演算処理ブロックEX0～EX3、汎用レジスタなどのレジスタブロックRB、バッファブロックBBから構成される。

【0026】命令コードISRは、例えば128ビットなどとされ、特に制限されないが、最大4つの処理フィールドf0、f1、f2、f3に分割されている。図1の例では、簡単のため、32ビットずつの均等の処理フィールドf0、f1、f2、f3を持つものとする。

【0027】命令リードブロックFUは、前記128ビットの命令を、図示されないROMやキャッシュメモリから、内部データバスIDBを介してリードして、格納し、更に、所定のタイミングで、命令デコードブロックDUに転送する。命令のリードアドレスは、プログラムカウンタに基づき、内部アドレスバスIABに出力される。命令のリード後、プログラムカウンタは更新される。命令リードブロックFUは、命令をリードする命令フェッチ部の一例である。

【0028】命令デコードブロックDUは、前記4つの処理フィールドf0、f1、f2、f3に対応した部分f0dec～f3decを含んで構成され、前記命令コードをデコードし、対応する演算処理ブロックEX0～EX3などに対応する制御信号CNT0～CNT3等を出力する。前記4つの処理フィールドf0、f1、f2、f3に対応した部分f0dec～f3decは、無操作を検出する部分f0nop～f3nopを含み、この検出結果によって、制御信号としてストップ信号STP0～STP3を出力する。また、レジスタブロックRB、バッファブロックBBへも図示を省略する制御信号を出力する。対応する処理フィールドf0～f3に、有効な処理の記述が含まれていない場合、或いは無操作を指示する記述が含まれている場合には、命令デコードブロックDUはストップ信号STP0～STP3により、対応する演算処理ブロックに動作停止の指示を与える。前記制御信号CNT0～CNT3は、ストップ信号STP0～STP3以外に、各演算処理ブロックEX0～EX3に個別に与えられる制御信号を総称している。前記命令デコードブロックDUは、前記命令フェッチ部がリードした命令に基づいて前記複数個の実行部の動作を制御する制御部の一例である。

【0029】演算処理ブロックEX0～EX3は、算術論理演算器（ALU）などを備え、命令デコードブロックDUの制御信号に基づいて、動作し、レジスタブロックRB、バッファブロックBBと内部バスDB0～DB3、WB0～WB3を介して、データの入出力を行い、所要のデータの演算処理を行なう。それぞれの演算処理ブロックEXi（i=0～3）は、命令デコードブロックDUの個別の制御信号に基づいて、独立した演算処理を行なう。前記演算処理ブロックEXiは、並列動作可能であって夫々演算器を内蔵した複数の実行部の一例である。

【0030】対応するストップ信号STPiがインエーブルにされた演算処理ブロックEXiは、クロック信号が停止され、データの入出力が禁止され、内部状態が保持される。演算処理ブロックEXiがパイプライン動作を行なう場合には、ストップ信号STPiは、パイプラインの動作に合わせて、伝播されるように構成する。この詳細な一例は図2に基いて詳述する。

【0031】レジスタブロックRBは、汎用レジスタ、コンディションコードレジスタなどを有し、所用のデータを、内部バスDB0～DB3、WB0～WB3を介して、各演算処理ブロックEXiに出力したり、演算結果を入力したりする。

【0032】バッファブロックBBは、アドレスバッファ、データバッファを備え、内部アドレスバスIAB、内部データバスIDBを介して、図示はされないRAMやキャッシュメモリとデータの入出力を行い、各演算処理ブロックEX0～EX3とのインタフェースを採る。

【0033】なお、命令コードのビット数、演算処理ブロックの数は、データ処理装置の応用範囲、回路規模、処理性能を勘案して最適な数値を選定すればよく、ここで示した数値は一例に過ぎない。例えば、内部バスは複数にしてもよい。内部アドレスバス／内部データバスは、命令コード用とデータ用を分離してもよい。

【0034】図1のCPU1によれば、命令コードに含まれる並列演算処理の情報を元にして、命令デコードブロックDUがストップ信号STPiによって、必要とされる演算処理ブロックEXiのみを動作させ、必要とされない演算処理ブロックEXiをストップ信号STPiにより自動的に停止することによって、不所望な電力消費を抑止することができる。

【0035】図2には演算処理ブロックEX0～EX3に含まれる演算器10のブロック図の一例が示される。同図に例示される演算器10は、セクタ11、算術論理演算器12、シフタ・結果バッファ13、ラッチ回路LT1～LT5、及びアンドゲートAND1～AND5から構成される。セクタ11は、内部バスDB0～DB3から入力が可能とされ、結果バッファ13は内部バスWB0～WB3に出力可能である。

【0036】演算器10は、後述の基準クロック信号φ

Bから生成されるノンオーバーラップの2相クロック信号φ1、φ2で動作する。

【0037】セクタ11は、クロック信号φ1の立ち上がりエッジに同期して、アンドゲートAND1を介して制御信号CNTiで指示されるデータを入力する。算術論理演算器12は、クロック信号φ1の立ち上がりエッジに同期して制御信号CNTiをラッチ回路LT1でラッチし、クロック信号φ1の1ステート（1サイクル）で演算を行なう。シフタ・結果バッファ13は、クロック信号φ1の立ち上がりエッジに同期して制御信号CNTiをラッチ回路LT2でラッチし、その出力をクロック信号φ2の立ち上がりエッジに同期してラッチ回路LT3でラッチし、ラッチされた制御信号をアンドゲートAND5を介して受けることにより、出力動作を行なう。

【0038】演算器10に供給されるストップ信号STPi（i=0〜3）は、クロック信号φ1、φ2の立ち上がりエッジに同期して順次ラッチ回路LT4、LT5にラッチされ、その伝播がクロック信号φ1、φ2に同期して順次遅延される。したがって、ストップ信号STPiがイネーブルレベル（この例ではハイレベル）にされると、先ず、クロック信号φ1の立ち上がりエッジに同期したセクタ11の入力動作が抑止され、次いで、クロック信号φ1の1ステートに同期される算術論理演算回路12の演算動作が抑止され、最後に、クロック信号φ2の1ステートに同期される結果バッファ13の出力動作が抑止される。

【0039】このように、制御信号CNTiの変化を抑止し、データの変化を抑止することによって、消費電力を抑止できる。CMOS半導体集積回路においては、実質的に、当該部分の消費電力を0にできる。演算器は高速で動作するように、消費電力が大きく、データの幅も32ビットなどと大きくなっており、更に、複雑な演算を行なうために、制御信号の本数も多くなっているから、かかる演算器のデータ及び制御信号の変化を抑止することによって、消費電力を低減することができる。

【0040】図2の構成より明らかなように、命令、即ち処理プログラムに対応して、1ステート単位、即ち基準クロック信号φBの単位で、低消費電力制御を行なうことができる。したがって、演算処理ブロックEXiがパイプライン動作を行なう場合に、換言すれば、1命令の演算実行ステージを1クロックサイクルで行なうとき、ストップ信号STPiは、パイプラインの動作に合わせて、伝播されるから、パイプラインを乱すことなく、若しくはパイプライン上の演算データを不所望に破壊することなく、命令コード単位で演算処理ブロックの並列度を制御して、低消費電力を図ることができる。

【0041】図3にはデータ処理装置の第2の構成例が示される。同図に示されるデータ処理装置もCPUとされる。図3に示されるCPU1Aは、命令リードブロッ

クFU、命令デコードブロックDU、4個の演算処理ブロックEX0〜EX3、リオーダバッファROR、リオーダバッファROM、汎用レジスタなどのレジスタブロックRB、及びバッファブロックBBから構成される。各演算処理ブロックEX0〜EX3は、リザーベーションステーションRSV0〜RSV3を含む。

【0042】図3のCPU1Aでは、ハードウェアによって並列処理の実行が制御される。即ち、命令デコードブロックDUは命令リードブロックFUの命令を解析して、並列処理可能な命令を抽出するデコーダアライメントユニットDAを持つ。命令デコードブロックDUは、デコーダアライメントユニットDAで抽出した命令を、並列にデコードして、制御信号CNT0〜CNT3、ストップ信号STP0〜STP3を各演算処理ブロックEX0〜EX3に与える。

【0043】各演算処理ブロックEX0〜EX3のリザーベーションステーションRSV0〜RSV3は、命令デコードブロックDUから出力された制御信号CNTi、STPiを一旦保持し、演算処理ブロックEX0〜EX3の実行状態を監視して、指示された処理の実行が可能になった時点で、演算処理を開始させる。

【0044】リオーダバッファROR、ROMは、プログラム上の命令の順序と整合性を保つように、並列実行している各演算処理ブロックEXiの演算結果のレジスタブロックRB、バッファブロックBBへの書き込みタイミングを調整している。

【0045】図3の例では、並列実行可能な命令が少ない場合には、命令デコードブロックDUはストップ信号STPiを、使用しない演算処理ブロックEXiに与える。このストップ信号STPiは、各演算処理ブロックEXiのリザーベーションステーションRSViで一旦保持され、実行中の処理が終了した時点で有効化される。ストップ信号STPiが有効化されると、演算処理ブロックEXiのクロック信号が停止され、データの入出力が禁止され、内部状態が保持される。演算処理ブロックEXiがパイプライン動作を行なう場合には、ストップ信号STPiが、パイプラインの動作に合わせて、伝播されるように構成されている点は、図1及び図2の場合と同様である。

【0046】このように、命令の実行状態に基づいて、ハードウェアで解析した並列処理情報を元にして、並列処理可能な数の演算処理ブロックEXiのみを動作させ、その他の演算処理ブロックEXiをストップ信号STPiによって、自動的に停止することによって、不所望な電力消費を抑止することができる。

【0047】更に、図3の例においては、並列度指定制御信号PALCが、命令デコードブロックDUに入力されている。かかる制御信号PALCに基づいて、命令デコードブロックDUは並列処理の制御時に、並列動作する演算処理ブロックEXiの数が前記制御信号PALC

10

20

30

40

50

によって指定される数を超えないようにする。前記制御信号P A L Cは、C P U 1 Aを採用するマイクロコンピュータ、マイクロプロセッサ、若しくはデータプロセッサ内の適宜のコントロールレジスタの設定値に従って生成される。そのようなコントロールレジスタへのデータ設定は例えばC P U 1 A自身がプログラムに従って行なえばよい。

【0048】図4には図3のC P U 1 Aによる命令実行手順を例示する。図3のC P U 1 Aは有意味の処理を行なう実行命令と、並列処理の制御を行なう制御命令を持つ。制御命令によって、続く命令列の並列実行の制御情報を生成し、命令デコードブロックD Uはこれを参照して、前記実行命令の並列実行を行なう。制御命令に従って、並列処理する数が少ない場合には、命令デコードブロックD Uは、前記の通り、所要のストップ信号S T P iをイネーブルにする。

【0049】前記制御命令は、有意味の処理ではないので、ソースプログラム上には記述を必要とせず、コンパイラで自動的に生成すれば都合がよい。

【0050】図5には本発明に係るデータ処理装置の第3の例であるシングルチップマイクロコンピュータが示される。

【0051】シングルチップマイクロコンピュータM P Uは、全体の制御を司るC P U 2 0、データトランスファコントローラ(D T C) 2 1、割込コントローラ(I N T) 2 2、バスコントローラ2 3、C P U 2 0の処理プログラムなどを格納するメモリであるROM 2 4、C P U 2 0の作業領域並びにデータの一時記憶用のメモリであるRAM 2 5、タイマ2 6、ウォッチドックタイマ2 7、シリアルコミュニケーションインタフェース(S C I) 2 8、A/D変換器2 9、入出力ポートI O P 1 ~ I O P 5、入出力ポートI O P A ~ I O P F、クロック発振器(C P G) 3 0、クロック制御回路3 1、及びブリスケーラ3 2から構成され、公知の半導体製造技術により1つの半導体基板(半導体チップ)上に形成される。C P U 2 0には、特に制限されないが、前述のC P U 1又はC P U 1 Aが採用されている。

【0052】前記シングルチップマイクロコンピュータM P Uは、電源端子として、グランドレベルV s s、電源電圧レベルV c c、アナロググランドレベルA V s s、アナログ電源電圧レベルA V c c、アナログ基準電圧V r e fの入力端子を有し、専用制御端子として、リセットR E S、スタンバイS T B Y、モード制御M D 0、M D 1、クロック入力E X T A L、X T A Lの各端子を有する。

【0053】C P G 3 0の端子E X T A L、X T A Lに接続される水晶発振子またはE X T A L端子に入力される外部クロック信号に基づいて、C P G 3 0は基準クロック信号(システムクロック信号)を生成する。基準クロック信号は、バスマスタの基準クロック信号φB、周辺

回路の基準クロック信号φSがある。シングルチップマイクロコンピュータM P Uの各機能ブロックは基準クロック信号φS、φBに同期して、動作を行う。

【0054】基準クロック信号φBは、バスマスタのクロック信号として、C P U 2 0、D T C 2 1、バスコントローラ2 3に、更に、外部バスのインタフェース用として入出力ポートI O P A ~ I O P Fに供給される。クロック信号φBはクロック制御回路3 1によって分周比が選択される。クロック信号φBの分周比が大きくなれば、これに反比例して、それに同期動作する部分の動作周波数は小さくなり、消費電力も低減される。

【0055】クロック信号φSは周辺クロック信号として、タイマ2 6、S C I 2 8、A/D変換器2 9、I O P A ~ I O P F、I O P 1 ~ I O P 5に供給される。クロック信号φSは周波数が固定とされているから、バスマスタの基準クロック信号φBが変更された場合も、不所望なクロック信号切替え処理などを必要としない。

【0056】クロック制御回路3 1から、D T C 2 1、タイマ2 6、ウォッチドックタイマ2 7、S C I 2 8、A/D変換器2 9に対して、クロック停止信号すなわちモジュールストップ信号M S T Pが与えられる。かかるモジュールストップ信号M S T Pは、クロック制御回路3 1から、各機能ブロックに独立した信号が与えられる。クロック停止時には、機能ブロックが内部状態を保持するようにされる。

【0057】ブリスケーラ3 2はクロック発振器3 0で発生されたクロック信号を分周して所要の回路に供給する。

【0058】シングルチップマイクロコンピュータM P Uの前記機能ブロックは、内部バスによって相互に接続される。内部バスは、アドレスバス、データバスの他に、リード信号、ライト信号、バスサイズ信号或いはシステムクロック信号などを伝達するコントロールバスを含む。内部データバスは、特に制限されないが、128ビット構成とされ、少なくともROM 2 4とC P U 2 0との間は、128ビットバスでインタフェースされる。特に制限はされないものの、RAM 2 5も同様に128ビットバスでインタフェースされる。内部アドレスバスはその位相によって、I A B、P A Bの2種類があり、内部データバスもその位相によって、I D B、P D Bが存在する。例えば、リードの場合、I A Bの後、P A Bは0.5ステート遅延する。P A BとP D Bは同期している。P D Bの後、I D Bは0.5ステート遅延する。I A BとP A B、I D BとP D Bは、バスコントローラ2 3によってバッファリングされている。

【0059】前記機能ブロックやモジュールは内部バスを介して、C P U 2 0によってリード/ライトさる。内蔵ROM 2 4、ROM 2 5は、I A B及びI D Bでインタフェースされ、1ステートでリード/ライト可能とされる。

【0060】前記シングルチップマイクロコンピュータMPUにリセット信号RESが与えられると、CPU20を始めとし、シングルチップマイクロコンピュータMPUはリセット状態になる。このリセットが解除されると、CPU20は所定のアドレスからスタートアドレスをリードして、このスタートアドレスから命令のリードを開始するリセット例外処理を行う。その後、CPU20は逐次、ROM24などから命令をリードし、解読して、その解読内容に基づいて演算処理を行う。

【0061】割込信号35は、A/D変換器29、タイマ26、ウォッチドックタイマ27、SCI28、入出力ポートIOP5が出力し、割込コントローラ22はこれを入力して、所定のレジスタなどの指定に基づいて、CPU20に割込要求信号36を与える。これにより、CPU20は実行中の処理を中断して、例外処理状態を経て、割込み要因に応答する所定の処理ルーチンに分岐し、所望の処理を行い、割込要因をクリアしたりする。機器制御においては、かかる割込み（イベント）の発生から、例外処理、所望の処理、割込み要因クリア等の処理を、所定時間内に処理するようにする必要がある。更に、考えられる複数のイベントの競合が発生した場合においても、かかる処理を、所定の時間内に処理する。したがって、通常、割り込み処理ルーチンでは、並列処理を最大にするような命令を配置するようにされる。所定の処理ルーチンの最後には、通常復帰命令が置かれ、この命令を実行することによって前記中断した処理を再開する。

【0062】図6には前記クロック制御回路31の一例が示される。クロック制御回路31は、分周器40、セクタ41、クロック制御レジスタ42、モジュールストップ制御レジスタ43、並列度指定制御レジスタ44、スタンバイ制御回路45によって構成される。並列度指定制御レジスタ44はCPU20に前記CPU1Aを採用する場合に必要な。CPU20の前記CPU1を採用する場合は不要である。

【0063】モジュールストップ制御レジスタ43はモジュールストップを選択可能な機能ブロックに対応して、所定のビット数分だけ設けられている。並列度指定制御レジスタ44は、少なくとも信号PALCによって通知可能な並列度数分のビット数を有する。

【0064】クロック制御レジスタは、SSBY、MSME、CKS1、CKS0の4ビットを有する。SSBYビットはスタンバイ状態への遷移を制御する。SSBYビットが1にセットされた状態で、CPU20がSLEEP命令を実行すると、スタンバイ状態に遷移し、SSBYビットが0にクリアされた状態で、CPU20がSLEEP命令を実行すると、スリープ状態に遷移する。スタンバイ状態では、シングルチップマイクロコンピュータMPU全体が停止するが、スリープ状態では、CPU20のみが停止する。

【0065】MSMEビットは中速モードの許可ビットであり、このビットが1にセットされている時のみ、CKS1、CKS0ビットが有効になる。

【0066】かかるMSME、CKS1、CKS0ビットの出力がセクタ41を制御し、 ϕOSC 、 $\phi OSC/2$ 、 $\phi OSC/4$ 、 $\phi OSC/8$ から、バスマスタのシステムクロック信号 ϕB を選択する。図7にはMSME、CKS1、CKS0によってシステムクロック信号 ϕB の選択態様が例示されている。なお、 ϕOSC 以外のクロック信号が選択された状態を中速モードと呼ぶ。

【0067】クロック発振器30には、EXTAL、XTAL端子が入力され、水晶振動子を接続するか、またはEXTALに外部クロック信号を入力する。これらと同一の周波数のクロック信号 ϕOSC を生成する。スタンバイ制御回路45から、発振器停止信号46が与えられ、スタンバイ状態ではクロック発振器30が停止し、クロック信号 ϕOSC はハイレベルで停止するようにされる。

【0068】分周器40は、クロック信号 ϕOSC を入力して、順次2分周を行って、クロック信号 $\phi OSC/2$ 、 $\phi OSC/4$ 、 $\phi OSC/8$ を生成する。

【0069】セクタ41は、クロック信号 ϕOSC 、 $\phi OSC/2$ 、 $\phi OSC/4$ 、 $\phi OSC/8$ を入力して、クロック制御レジスタ42のMSMEビット、CKS1、CKS0ビットで選択されたクロック信号を、システムクロック信号 ϕB として出力し、シングルチップマイクロコンピュータMPUのCPU20等のバスマスタに供給する。

【0070】クロック制御レジスタ42、モジュールストップ制御レジスタ43及び並列度指定制御レジスタ44は、内部データバスPDBに接続され、CPU20からリード/ライト可能にされている。

【0071】クロック制御レジスタ42は、MSMEビット、CKS1、CKS0ビットを、セクタ41に出力するとともに、SSBYビットをスタンバイ制御回路45に出力する。モジュールストップ制御レジスタ43は、モジュールストップ信号MSTPを、各機能ブロックに供給する。

【0072】スタンバイ制御回路45は、例えば、フリップフロップで構成され、SSBYビットが1の状態では、CPU20がSLEEP命令を実行すると、かかるフリップフロップがセット状態にされ、発振器停止信号46を活性状態とし、これによってマイクロコンピュータMPUをスタンバイ状態とする。スタンバイ状態で外部割り込み要求が発生すると、例えば、割り込みコントローラ22の制御によって、スタンバイ解除信号S1が活性状態になり、前記フリップフロップはリセット状態になり、発振器停止信号46が非活性状態になり、スタンバイ状態が解除される。

【0073】図8には前記シングルチップマイクロコン

ビュータMPU若しくはCPU1, CPU1Aの開発環境の概略が示される。シングルチップマイクロコンピュータMPUの使用者は、各種エディタなどを用いて、C言語乃至アセンブリ言語でプログラムを作成する。これは通常、複数のモジュールに分割して作成される。

【0074】Cコンパイラ50は、使用者の作成したそれぞれのC言語ソースプログラムを入力し、アセンブリ言語ソースプログラム乃至オブジェクトモジュールを出力する。Cコンパイラ50の処理においては、図示はされないプリプロセッサの処理が行われる。

【0075】アセンブラ51は、アセンブリ言語ソースプログラムを入力し、オブジェクトモジュールを出力する。

【0076】リンカージェディタ52は、上記Cコンパイラ50やアセンブラ51の生成した、複数のオブジェクトモジュールを入力して、各モジュールの外部参照や相対アドレスなどの解決を行い、1つのプログラムに結合して、ロードモジュールを出力する。

【0077】ロードモジュールは、シミュレータデバッガ53に入力され、図示を省略するパーソナルコンピュータなどのシステム開発装置上で、シングルチップマイクロコンピュータMPUの動作をシミュレーションし、実行結果を表示し、プログラムの解析や評価を行なうことができる。また、エミュレータ54に入力して、実際の応用システム上などで動作する、所謂インサーキットエミュレーションを行ない、マイクロコンピュータ全体としての、実動作の解析や評価を行なうことができる。

【0078】このほかに、ライブラリアンとして、汎用的なサブルーチンなどを提供することもできる。

【0079】図9には前記シングルチップマイクロコンピュータMPUのシステム開発装置におけるCPU1 (CPU1A)の並列処理選択方法を示す。例えば、図9の(a)に示される通り、パーソナルコンピュータなどのシステム開発装置上のディスプレイでプロンプトが表示された状態で、例えば、SET PARALLEL=4と入力すればよいようにする。或いは、(b)に示されるように、プロンプトが表示された状態で、SET PARALLELとコマンドを入力し、これに対して、CPUの種類及び動作モードのメニューを、例えばPARALLELNo. (1. PARALLEL:1, 2. PARALLEL:2, 3. PARALLEL:3, 4. PARALLEL:4)と表示するとともに、メニュー番号の入力を要求し、使用者がメニューの番号1~4のいずれかを入力すればよいようにする。この後コンパイルされるファイルは上記で設定した並列数が使用される。

【0080】また、(c)に例示されるように、Cコンパイラを実行する場合のオプションとして、並列数を設定してもよい。例えば、ccomp-PARALLEL=4 source. cと入力する。オプション「-PA

RALLEL=4」で4並列を指定し、C言語ソースプログラムファイル「source. c」をコンパイルする。

【0081】このほか、ウィンドウのドロップダウンメニューで選択可能にしてもよいし、ワークステーションなどであれば、Cシェルコマンドとして入力することもできる。

【0082】更に、アセンブラやCコンパイラなどの、ソースプログラムの制御命令として、CPU1の並列処理の数及び動作モードを入力することができる。

【0083】並列数を少なくした場合には、図1のCPU1に対応する開発環境では、命令コードの一部が常に、演算処理を示さないようにされる。

【0084】図3のCPU1Aに対応する開発環境では、Cコンパイラなどで、並列処理しやすい命令の順序を生成するが、並列数を少なくすれば、その分、Cコンパイラの処理が少なくなるので、コンパイル時間の短縮などに寄与できる。また、並列処理制御命令を生成する場合にも同様である。

【0085】また、並列処理数の異なるCPU (データ処理装置)を展開した場合にも、Cコンパイラなどの開発装置を共通に利用可能にでき、開発環境の開発効率を向上することができる。

【0086】図10にはCPU1, CPU1Aのソースプログラムにおける並列処理選択方法を示す。

【0087】C言語のソースプログラムは、コンパイル処理前に、プリプロセッサの処理を受ける。ソースプログラムでは、「#」で示されるプリプロセッサ文によって、プリプロセッサは処理を行い、コンパイラに指示が行われる。

【0088】図10では、C言語で記述されたソースプログラムであり、プリプロセッサ文(#pragma)で、コンパイラの処理を指示することができる。(a)の場合、

```
#pragma parallel 4 (kansul)
#pragma parallel 2 (kansu2)
で、関数「kansul」は4並列で、関数「kansu2」は2並列でコンパイルすることが指示される。
```

【0089】また、(b)の場合、

```
#pragma parallel 2
```

...

```
#pragma parallel 2 end
```

で、括られた記述(この場合は、whileループ)が、2並列でコンパイルされる。

【0090】図1のCPU1のように、128ビットの命令コードに、4つの処理内容が記述される場合、2並列でコンパイルした場合には、所定の2つの演算処理ブロックに対応するフィールドが無操作となる。無操作のフィールドは、命令デコードブロックDUによって、対応する演算処理ブロックにストップ信号STPiをアサ

ートさせ、その演算処理動作を停止状態にする。」

【0091】また、図3のCPU1Aのように、並列処理を制御する命令を生成する場合には、2並列でコンパイルすると、3以上の並列処理が指示されないようにされる。

【0092】図11には前記シングルチップマイクロコンピュータMPUをプリンタ制御に用いたマイクロコンピュータシステムが例示される。

【0093】プリンタ制御システムは、シングルチップマイクロコンピュータMPU、セントロニクスインタフェース或いはユニバーサルシリアルバスなどの受信回路60、バッファRAM (DRAM) 61、キャラクタジェネレートROM (CGROM) 62、プログラムROM 63を含み、これらがシングルチップマイクロコンピュータMPUの外部バス64を介して接続される。

【0094】また、図11のプリンタシステムは、更に、印字ヘッド65、バッファ回路66、ラインフィードモータ67、及びキャリッジリターンモータ68を含み、これらのモータ67、68は、それぞれタイマ26の出力等によって制御される。ラインフィードモータ67及びキャリッジリターンモータ68は、特に制限はされないものの、ステッピングモータである。

【0095】受信回路60から入力されたデータは、一旦、バッファRAM 61に格納された後、CPU 20によって展開され、バッファRAM 61から所定の順序で印字ヘッド65に送られる。この処理量は、印刷するデータによって相違される。例えば、テキストデータはデータ量が比較的少なく、CPU 20の処理量も比較的小さい。一方、画像データは、データ量が多く、CPU 20の処理量も多い。

【0096】従って、図11のシステムにおいて、テキストデータを扱うCPU 20の処理プログラム (関数) は並列度を低くし、消費電力を低減し、画像データを扱う処理プログラム (関数) は並列度を高くするとよい。並列度を変更することによって、所望の処理能力を維持しつつ、不所望な電力消費を抑止できる。

【0097】印字ヘッドデータを転送する場合、印字ヘッド65は印刷精度を向上するため、不規則な形状をしていたり、種々の工夫が試みられ、変更される場合も多いから、DTC 21などの固定的なデータ転送では対応が困難で、印字ヘッド65の仕様に合わせて、CPU 20がデータを転送することが必要な場合がある。

【0098】この場合、CPU 20の処理能力によって、印字性能が決る可能性がある。換言すれば、所要の印字性能を実現する為には、上記のデータ転送処理を、所定時間内に終了する必要がある。本発明においては、1つの処理プログラムを、並列の演算器を用いて、高速化しているから、かかる制御に好適である。

【0099】図5のシングルチップマイクロコンピュータMPUを用いる場合には、CPU 20の処理負荷を軽

減するために、受信回路60からバッファRAM 61へのデータ転送を、CPU 20の動作と並行して、DTC 21が行ったりすることができる。また、DTC 21は、ラインフィードモータ67及びキャリッジリターンモータ68を駆動するパルス出力を行う。更に、SCI 28の送信データ、受信データの転送を行なう。

【0100】図11のプリンタシステムにおいて、図示はされないものの、SCI 28はホスト装置などとの通信に使用され、A/D変換器29は紙枚数などのセンサ情報を入力する。タイマ26は、上記モータ駆動の基準タイミングを発生する。これらの基準クロック信号φSは固定であるから、これらの動作状況に拘らず、CPU 20などのバスマスタの基準クロック信号φBを変更して、随時消費電力を低減することができる。モータの駆動信号は、タイマ26の基準タイミングに従うから、DTC 21によるデータ転送に必要な時間が増えても、基準タイミング周期内であれば問題ない。

【0101】周知のように、最近では、プリンタなどのAC電源を使用するものにあっても、消費電力の低減が望まれている。周辺機能の内、使用しないものは、モジュールストップ信号MSTPによって、動作を停止させ、消費電力を低減することができる。

【0102】半導体集積回路の集積度の向上によって、受信回路60の一部をシングルチップマイクロコンピュータMPUに搭載して1個の半導体集積回路化することができる。集積度の向上により、配線容量が低減できるから、消費電力の低減にも寄与できる。更に、バッファRAM 61などの汎用的なメモリも1個の半導体集積回路MPUに集積することができる。プログラムROM 62やCGROM 63などのように個別のプリンタの機種に応じて変更になるものは、個別の半導体集積回路にする方が都合がよい。

【0103】図12には前記マイクロコンピュータMPUのためのエミュレータが例示される。図12において70は図11のプリンタシステムのような応用システム (ターゲットシステム又はユーザシステムとも称する)、71はエミュレータ、72はシステム開発装置である。

【0104】エミュレータ71はエミュレーション用プロセッサ73を有する。エミュレーション用プロセッサ73は、応用システム70に用いられるマイクロコンピュータMPUと同等のマイクロコンピュータ部分にエミュレーション用インタフェースを加えて構成される。エミュレーション用プロセッサ73の前記マイクロコンピュータ部分はインタフェースケーブル74を介して、応用システム70のマイクロコンピュータMPUの実装ソケット75に結合され、これによってエミュレーション用プロセッサは応用システムを代行制御することができる。

【0105】一方、エミュレーション用プロセッサ73

は前記エミュレーションインタフェースを用いてエミュレーションバス76に接続される。エミュレーションバス76のデータバス幅は、1DBのバス幅に対して128ビット等とされる。エミュレーションバス76には図示はされない状態信号・制御信号の信号線などを含む。前記エミュレーションバス76を用いて、エミュレーション用プロセッサ73から、応用システム70とエミュレーション用プロセッサ73の内部状態に応じた情報などが出力され、また、エミュレーション用プロセッサ73に対し、エミュレーションのための各種信号が入力される。

【0106】さらに、前記エミュレーションバス76には、エミュレーションメモリ81、ブレーク制御回路82、リアルタイムトレース回路83などが接続される。前記エミュレーションメモリ81は、特に制限はされないものの、RAMなどによって構成され、前記のユーザプログラムを格納した領域と、エミュレーションのためのプログラムを格納した領域とを持つ。前記ブレーク制御回路82は、エミュレーション用プロセッサ73による制御状態やエミュレーションバス76の状態を監視して、その状態が予め設定された状態に達した時に、ブレーク割込み信号BRKをアサートして、エミュレーション用プロセッサ73のCPU20によるユーザプログラムの実行を停止させ、エミュレーション用プログラム実行状態に遷移させる（ブレークする）。前記リアルタイムトレース回路83は、前記CPU20のリード動作またはライト動作を示す信号、命令リード動作を示す信号（CPUステータス信号）、エミュレーションバス76に与えられるアドレスやデータさらには制御信号を逐次蓄える。

【0107】前記エミュレーションメモリ81、ブレーク制御回路82、リアルタイムトレース回路83はコントロールバス84にも接続され、コントロールバス84を介してコントロールプロセッサ85の制御を受けるようになっている。前記コントロールバス84は、前記コントロールプロセッサ85に接続されるとともに、ホストインタフェース回路86を介して、特に制限はされないものの、前記パーソナルコンピュータなどのシステム開発装置72に接続される。

【0108】例えば、システム開発装置72から入力されたプログラム（ロードモジュール）をエミュレーションメモリ81のユーザプログラム格納領域に転送し、内蔵ROM上に配置されるべきかかるプログラムをCPU20がリードすると、エミュレーションメモリ81上のプログラムがリードされ、実行される。また、ブレーク条件や、リアルタイムトレース条件などもシステム開発装置72から与えることができる。

【0109】前記リアルタイムトレース回路83は、エミュレーションバス76のアドレス、データ、制御信号、CPUステータス信号などを蓄積し、この内容は、

所定のコマンドなどでシステム開発装置72の表示装置に表示して、使用者は、この内容に基づいて、プログラムの動作解析を行なう。前記の通り、CPU20のフェッチした命令は、エミュレーションバス76上に現われ、これを逆アセンブルした、アセンブリ言語でも表示することができる。更に、C言語ソースプログラムも表示することができる。

【0110】前述の通り、CPU20は単一の命令ストリームを実行するようになっているから、ブレークの設定や、トレース情報の取得など、エミュレータ71による応用システム70上での動作確認が容易である。また、図1に示されるCPU1のように、命令コード上に、並列動作の情報が含まれる場合、その情報は、トレース情報として、トレースできるから、動作解析に更に好適である。

【0111】上記実施例によれば、以下の作用効果を得るものである。

【0112】（1）実行する並列処理の数に対応して、不必要な演算処理ブロックEXIを停止状態にすることにより、不所望な電力消費を抑止することができる。消費電力を低減することにより、不要輻射即ち高周波ノイズを低減することができる。

【0113】（2）Cコンパイラなどで、命令コードによって、並列度を指定可能にすることで、CPUの処理状態に応じて、処理性能を優先するか、消費電力抑止を優先するかの選択が可能になる。

【0114】（3）並列度指定制御レジスタ44のような内部I/Oレジスタによって、並列度を指定可能にすることによって、データ処理装置の処理状態に応じて、処理性能を優先するか、消費電力抑止を優先するかの選択が可能になる。

【0115】（4）クロック制御レジスタ42のような内部I/Oレジスタによって、クロック信号を変更することにより、プログラムを変更することなく、CPUの処理状態に応じて、処理性能を優先するか、消費電力抑止を優先するかの選択が可能になる。これは、命令コードで並列度が指定される場合にも利用できる。一つの関数などが、処理速度が必要な場合と、低消費電力が必要な場合の両方に使用される場合にも、呼び出し側で、クロック信号を変更することによって、処理性能を優先するか、消費電力抑止を優先するかの選択が可能になる。共通の関数とすることにより、プログラムの開発効率を向上できる。周辺クロック信号とは独立に、CPUなどのバスマスタクロック信号の分周比を可変にすることによって、周辺回路に対する不所望の処理を必要とせず、使い勝手を向上することができる。

【0116】（5）Cコンパイラなどの開発環境において、並列度を指定可能にすることによって、並列度が異なるデータ処理装置についても開発環境を共通に利用可能にできる。開発環境の開発効率を向上できる。並列度

の拡張を考慮して設計しておく、乃至、予め、上記拡張に対応させておくことにより、更に、開発効率を向上することができる。

【0117】(6)Cコンパイラにおいてステップ単位、関数単位、ソースプログラムのファイル単位などで、並列度を指定することによって、より使い勝手を向上することができる。

【0118】以上本発明者によってなされた発明を実施形態に基づいて具体的に説明したが、本発明はそれに限定されるものではなく、その要旨を逸脱しない範囲において種々変更可能であることは言うまでもない。

【0119】例えば、並列処理の数は任意にできる。演算処理ブロックは全て同じ機能である必要はない。1ステート演算用、ロード／ストア処理用など個別の処理機能を持たせてもよい。並列処理の方法も、VLIWやスーパースカラなどのほか、MIMD(マルチプル・インストラクション・マルチプル・データ)などであってもよい。

【0120】データ処理装置の詳細、並列処理の制御の方法や、演算処理ブロックの詳細も、任意に変更可能である。

【0121】また、命令コードの基本単位128ビットに限定する必要はなく、いわゆるVLIW型のデータ処理装置にあっては、256ビットや64ビットでもよく、いわゆるスーパースカラ型のデータ処理装置にあっては、16ビット或いは32ビットなど任意のビット幅とできる。データバスの幅も任意にできる。

【0122】シングルチップマイクロコンピュータのその他の機能ブロックについても何等制約されない。データ処理装置の実現方法としては、シングルチップマイクロコンピュータや半導体集積回路には限定されない。

【0123】マイクロコンピュータシステムとしてもプリンタに限定されない。携帯型機器、例えばオートフォーカスのカメラなどであってもよい。焦点を合わせるまでは並列度を高くし、一旦焦点が合った後の被写体の動きに焦点を追従させるときには並列度を低くすることが考えられる。

【0124】開発環境の詳細についても、任意にできる。少なくとも、ユーザがプログラムを作成する際に、並列度を指定できるようにすればよい。

【0125】以上の説明では主として本発明者によってなされた発明をその背景となった利用分野であるシングルチップマイクロコンピュータに適用した場合について説明したが、それに限定されるものではなく、その他のマイクロコンピュータやデータプロセッサなどその名称の如何に拘わらずデータ処理装置に広く適用可能である。

【0126】

【発明の効果】本願において開示される発明のうち代表的なものによって得られる効果を簡単に説明すれば下記

の通りである。

【0127】すなわち、並列処理を行なう場合に、演算処理ブロックの数より、並列実行可能な処理が少ない場合に、動作を要しない演算処理ブロックのクロック信号を停止させると共に、データ入出力などを禁止することにより、不所望な電力消費を抑止することができる。

【0128】動作許可する並列処理の数を、コンパイラや制御レジスタで、指定可能にすることによって、一部の並列処理ブロックの動作を停止させて、消費電力を低減することができる。

【0129】制御レジスタでクロック信号を変更することによって、プログラムを変更することなく、データ処理装置の処理状態に応じて、処理性能を優先するか、消費電力抑止を優先するかの選択が可能にできる。

【図面の簡単な説明】

【図1】本発明に係るデータ処理装置の第1の例としてCPUを例示するブロック図である。

【図2】図1のCPUに含まれる演算処理ブロックの演算器を例示するブロック図である。

【図3】本発明に係るデータ処理装置の第2の例として別のCPUを示すブロック図である。

【図4】図3のCPUによる命令実行手順を例示する説明図である。

【図5】本発明に係るデータ処理装置の第3の例としてシングルチップマイクロコンピュータを例示するブロック図である。

【図6】シングルチップマイクロコンピュータに内蔵されるクロック制御回路を例示するブロック図である。

【図7】MSME、CKS1、CKS0によるシステムクロック信号φBの選択態様を例示する説明図である。

【図8】CPUの開発環境を概略的に示す説明図である。

【図9】CPUのシステム開発装置における並列処理選択方法を示す説明図である。

【図10】CPUのソースプログラムにおける並列処理選択方法を示す説明図である。

【図11】図5のシングルチップマイクロコンピュータをプリンタ制御に用いたマイクロコンピュータシステムのブロック図である。

【図12】図5のマイクロコンピュータのためのエミュレータを例示するブロック図である。

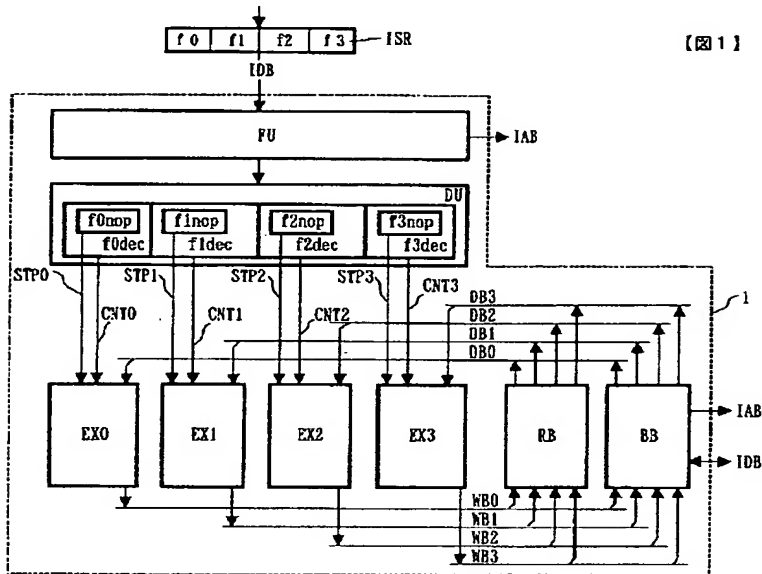
【符号の説明】

1, 1A, 20 CPU
FU 命令リードブロック
DU 命令デコードブロック
DA デコーダアライメントユニット
EX0~EX3 演算処理ブロック
RSV0~RSV3 リザーベーションステーション
RB レジスタブロック
BB バッファブロック

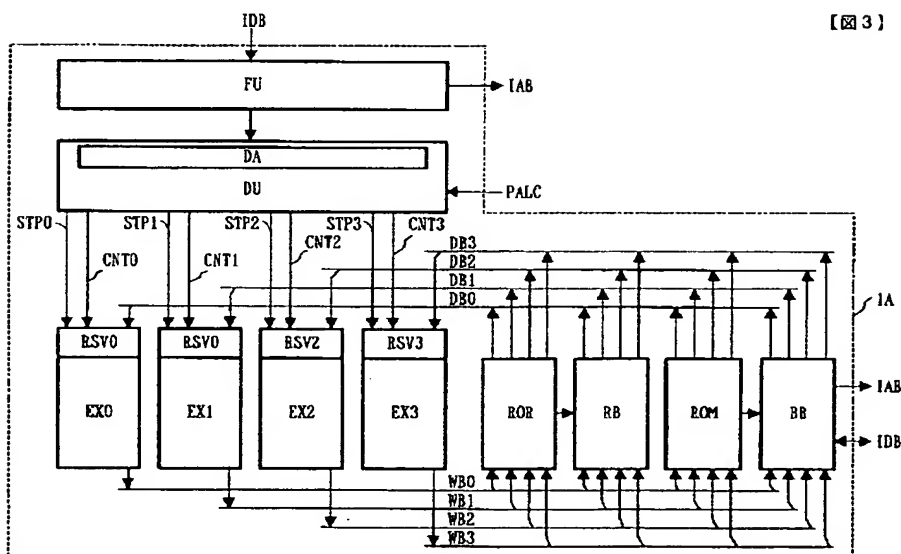
STP0~STP3 (STP_i) ストップ信号
 CNT0~CNT3 (CNT_i) 制御信号
 $\phi 1$ 、 $\phi 2$ ノンオーバーラップ2相クロック信号
 ϕB 、 ϕS 基準クロック信号
 PALC 並列度指定制御信号
 10 演算器

* 1 1 セレクタ
 1 2 算術論理演算回路
 1 3 シフト回路・結果バッファ
 3 0 クロック発振回路
 3 1 クロック制御回路
 * 4 4 並列度指定制御レジスタ

【図1】



【図3】



【図7】

【図7】

MSME	CKS1	CKS0	ϕB
0	-	-	ϕOSC
1	0	0	ϕOSC
	0	1	$\phi OSC/2$
	1	0	$\phi OSC/4$
	1	1	$\phi OSC/8$

【図10】

【図10】

(a)

```

void kausul(void);
void kausu2(void);
#pragma parallel14(kausul)
#pragma parallel12(kausu2)
...
void kausul(void)
{
    ...
}
void kausu2(void)
{
    ...
}
...

```

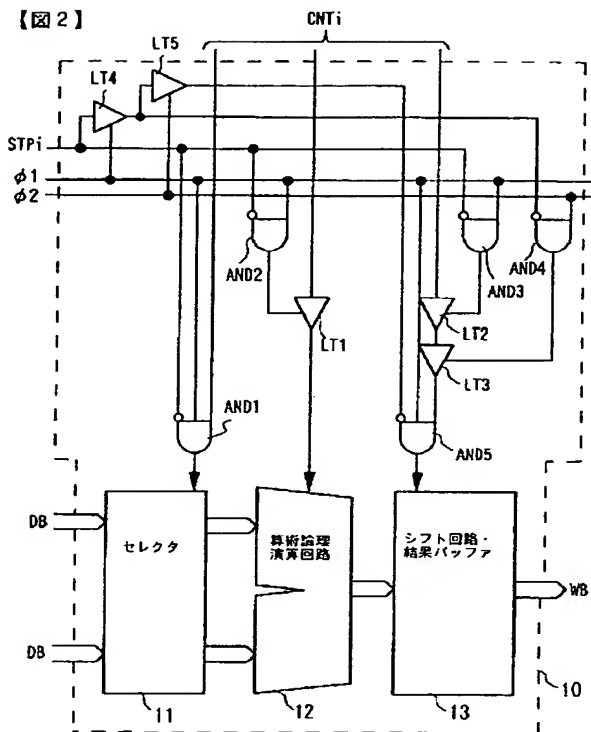
(b)

```

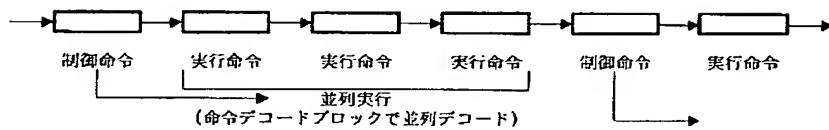
void kausul(void);
void kausu2(void)
{
    ...
    #pragma parallel12
    while (...){
        ...
    }
    #pragma parallel12end
    ...
}

```

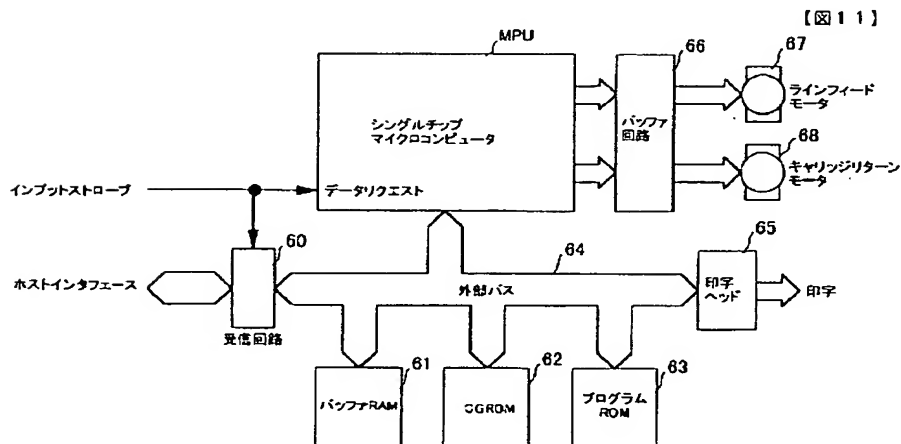
【図2】



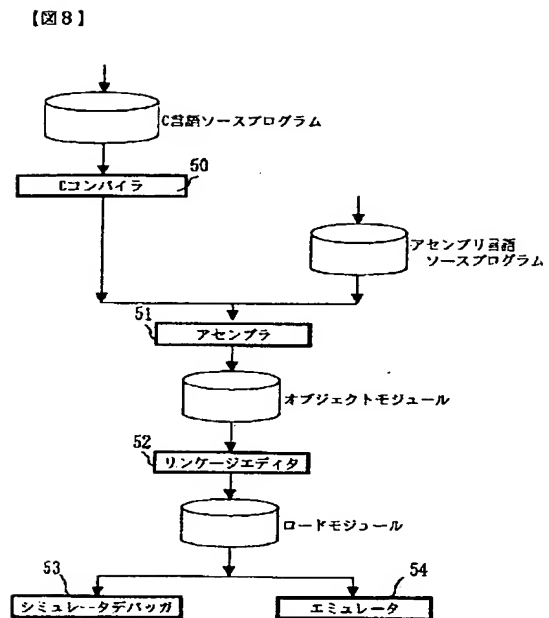
【図4】



【図11】

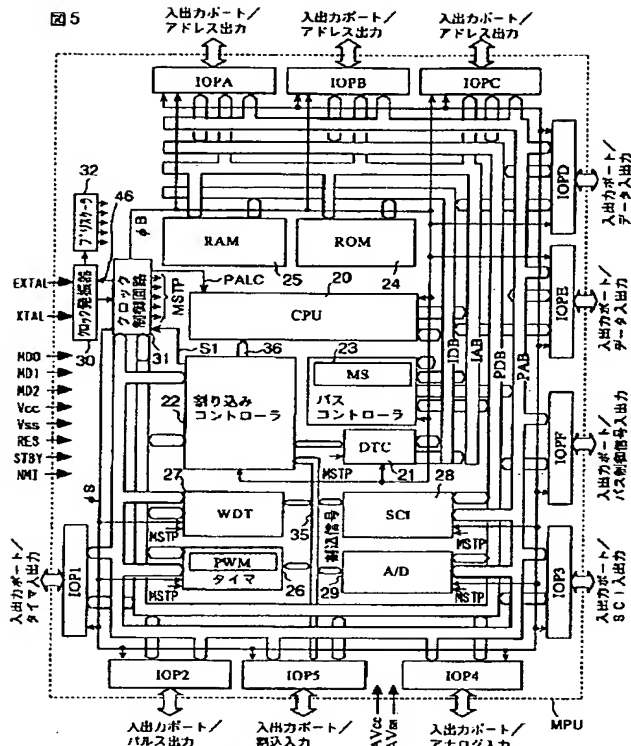


【図8】

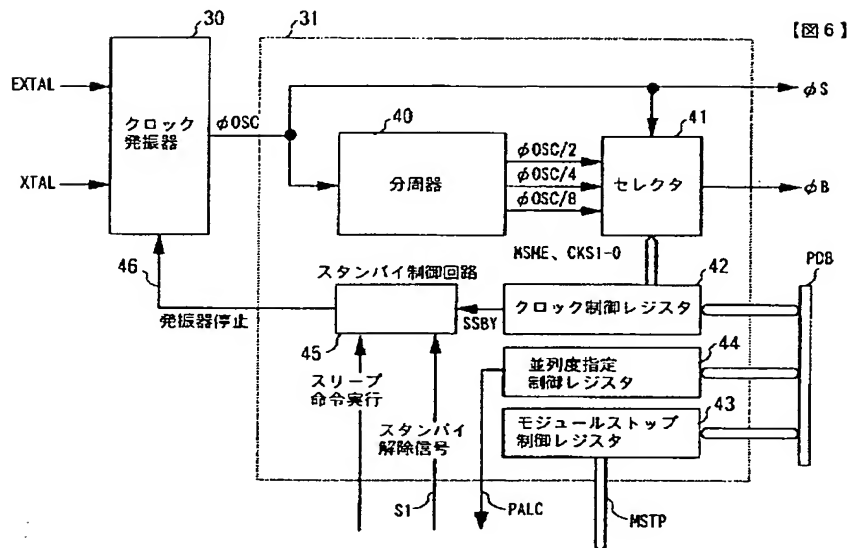


【図4】

【図5】

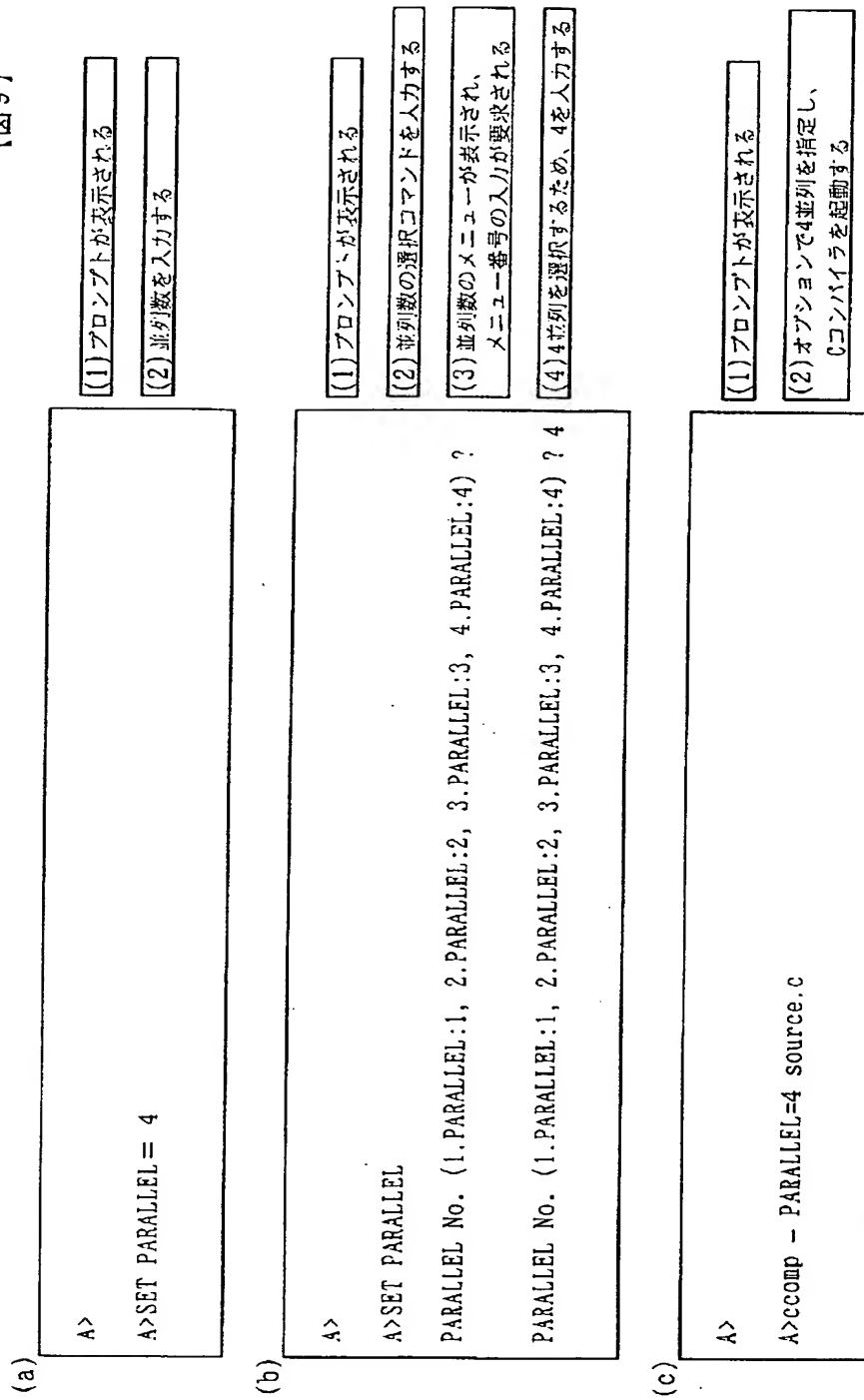


【図6】

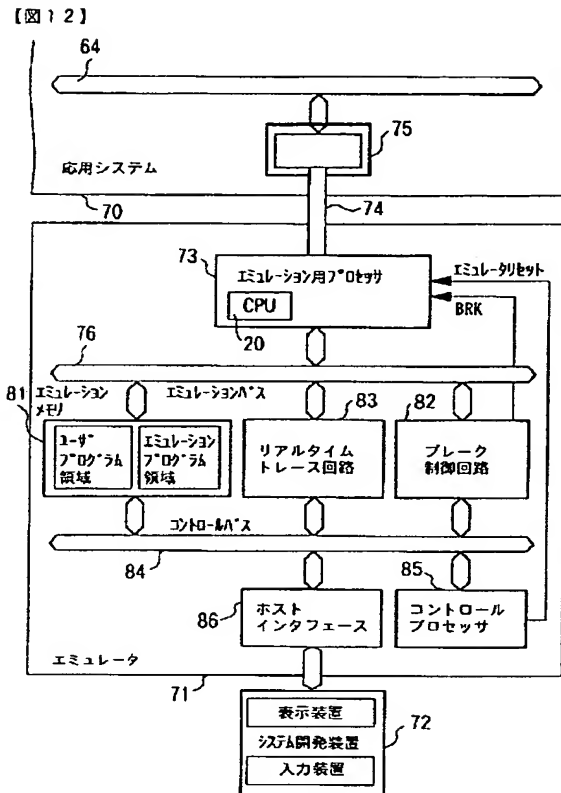


【図9】

【図9】



【図12】



フロントページの続き

(51)Int.Cl.⁷
G 0 6 F 9/45

識別記号

F I
G 0 6 F 9/44

テーマコード (参考)
3 2 2 F

F ターム (参考) 5B011 DA06 EA08 EA09 LL08 LL11
LL12
5B013 DD01 DD04
5B033 AA05 AA13 AA14 BC01 BD01
BE05 DD05
5B079 AA07 BA03 BA12 BA13 B801
BC01 DD03
5B081 AA06 AA07 CC32